Binôme : Amarouche Lies et veronique Huang

Projet 12 : boite a outils

Introduction

Nous avons décidé de prendre le « projet 12 : la boite a outils ». Ce projet en SDL sur la modification d'image nous intéressait/intriguait particulièrement sur le fait de savoir comment fonctionnait les modification d'image sur les images comment accéder au pixels ? qu'es ce qu'une image ? comment la parcourir ?

1 /Résumé :

Pour commencer, nous avions chacun installer SDL 2.0 sur nos ordinateur en y ajoutant la librairie SDL. Nous avions ensuite fait des recherches pour comprendre ce langage et ainsi avoir les bases. Peu à peu avec nos recherches nous avions pu afficher correctement une image dans une fenêtre. Après cette étape, nous avions compris que c'était grâce aux pixels de notre image(surface) que nous allions la modifier pour changer sa couleur, sa luminosité, son contraste… Ainsi pour accéder aux pixels qui sont sur un tableau a deux dimension il suffisais de les parcourir avec deux boucles for imbriqué on parcours d'abord la hauteur (colonnes) ensuite la largeur(lignes) et utiliser des pointeurs (sur un objet) sur l'image qui nous permettait d'accéder à la valeur d'un pixel, après pour avoir les coordonnes de chaque pixel on a utilisé SDL_GetRGB et pour appliquer les modifications on a utilisé SDL_MapRGB avec en paramètre l'adresse de la valeur format de l'image et 3 valeurs non signé sur 8 bits. Ainsi pour le format de l'image il suffit d'écrire image->format (tel que l'image est une surface) et les 3 autres paramètres dépendait de quel effet nous voulions, pour le gris, la luminosité ou bien le contraste, …il suffit de changer la valeur du Rouge, du Vert et du Bleu.

2/Programmes et fonctions :

2.1/programmes :

Nous avons fait un programme qui fonctionne par events (effects.c), il englobe toutes les fonctions que nous avons mis en place, et prennent effet quand on clique sur une touche du clavier. Ensuite on a aussi fait des programmes qui fonctionnent en ligne de commande de la sorte que chaque programme a son propre fonctionnement et correspond à un effet sur une image.

Tous nos programmes fonctionnent qu'avec des images en formats bmp.

2.2/description des fonctions :

2.2.1/fonctions SDL utilisés :

SDL_CreateWindow : elle permet de créer une fenêtre elle prend comme argument la largeur et la longueur de la fenêtre le titre de la fenêtre notamment.

SDL_CreateRendrer : elle permet de créer un rendu sur la fenêtre elle prend comme argument la fenêtre, l'index du rendu qu'on met généralement a -1 et un drapeau qui est Uint32 on met generalemnt sa valeur a 0.

SDL_CreateWindowAndRendrer : elle permet de créer de le rendu et la fenêtre en même temps elle prend en paramètre l'adresse de la fenêtre et du rendu et la taille de la fenêtre .

SDL_INIT_VIDEO : elle permet d'initialiser la SDL.

SDL_LoadBMP : elle permet de charger une surface (une image) .

SDL_CreateTexture : elle permet de créer une texture à partir du rendue elle prend comme argument le rendu, le format, l'Access, a largeur et sa hauteur comme valeur pour le format et l'accès on met les valeurs de **SDL_PixelFormatEnum et SDL_TextureAccess .**

SDL_UpdateTexture : elle permet de mettre à jour la texture avec de nouvelles données par exemple quand on modifie les pixels.

SDL_ConvertSurface : elle permet de copier une surface dans une nouvelle avec son propre format .

SDL_RenderReadPixels : elle lis les pixels du rendu et les sauvegardes dans une surfaces

SDL_SaveBMP : elle permet de sauvegarder une surface.

SDL_CreateRGBSurface : elle permet de créer une nouvelle surface RGB elle prend en argument un drapeau qu'on met généralement a 0 la largeur et la hauteur de la surface la profondeur qu'on met généralement a 32 et les masks.

SDL_RenderCopy : elle permet d'afficher la texture a partir du rendu .

SDL_RenderPresent : elle permet dafficher le rendu a l'ecran.

2.2.2/exitwe :

La fonction exitwe est présente dans tous nos programmes on la mise en place pour prendre en charge les retours de fonctions, elle renvoie un message d'erreur sur la console.

2.2.3/grayscale :

La fonction a pour but de transformer une image couler en une image en noir et blanc, dans ce programme on parcourt l'image pixel par pixel ensuite en extrait sa valeur rouge,vert,bleu de chaque pixel grâce à SDL_GetRGB et on crée une variable Uint8 qui va recevoir a chaque fois 0.212671f du rouge initiale , 0.715160f du vert et 0.072169f de bleu et on applique les nouvelles valeur au pixel grace a SDL_MapRGB.

La fonction est présente dans deux programmes effects.c elle prend effet quand on clique sur la touche g et si on veut sauvegarder le résultat on doit just ajouter un deuxième argument au programme avant de le lancer. Elle est aussi présente dans le programme gray.c qui fonctionne en ligne de commande il suffit de lancer le programme avec deux arguments l'image source et l'image destination on retrouve le résultat dans l'image destination.

2.2.4/ Negatif:

La fonction a pour but d'inverser les pixels d'une image, dans ce programme on parcourt l'image pixel par pixel ensuite en extrait sa valeur rouge,vert,bleu de chaque pixel grâce à SDL_GetRGB on soustrait a chaque pixels la valeur 255 grâce SDL_MapRGB.

La fonction a le même fonctionnent que grayscale sauf que dans le programme effects.c on appuis sur la touche n au lieu de la touche g.

2.2.5/ luminosite:

Cette fonction utilise une fonction auxiliaire qu'on a appelé fluminosite , cette fonction englobe la définition mathématique de la luminosité qui est d'ajouter de luminosité c'est-à-dire du blanc pour éclaircir et du noir pour assombrir , dans la fonction luminosite on parcourt l'image pixel par pixel et on applique à chaque pixel fluminosit , cette fonction prend en compte l'intensité de la luminosité qui est un entier. Si n=0 rien ne change ,plus n > 0 plus l'image s'éclaircit et plus n < 0 plus l'image s'assombrit .

2.2.6/ contrsate:

Cette fonction utilise elle aussi une fonction auxiliaire qu'on appelé fcontraste , cette fonction englobe la définition mathématique du contraste ; rendre les pixels lumineux encore plus lumineux et sombre encore plus sombre , quant a la fonction contraste c'est la même que luminosité sauf qu'on applique la fonction fcontraste.

2.2.7/ flou :

Cette fonction utilise une fonction auxiliaire qu'on a appelé moyenne, son rôle principal est de remplacer un pixel par la moyenne des pixels qui l'entourent, et puisque on utilise les valeurs d'autre pixels on doit créer un nouveau tableau de pixels, dans la fonction on parcourt l'image comme d'habitude et on applique la fonction moyenne à chaque pixel à chaque itération.

2.3/description des programmes :

2.3.1/ effects.c :

Ce programme prend en compte deux arguments l'image source(l'image qu'on veut afficher) et l'image destination(optionnel) , en lui on retrouve toutes les fonctions cites ci-dessus c'est un programme qui marche avec les évènements du clavier ; on a créé une fenêtre avec un rendu et une texture pour afficher l'image dessus et pour ensuite appliquer les modifications qu'on veut on a utilisé SDL_UpdateTexture pour actualiser la texture à chaque fois et SDL_ConvertSurface pour actualiser la surface, ce programme s'arrête qu'on clique sur quitter avec la souris ; pour changer l'image en noir et blanc on clique sur le touche « g », pour la rendre négative on clique sur la touche « n », pour le flou ça sera la touche « f » , pour la rotation qu'on a mis en place grâce à SDL_RenderCopyeEX ça sera « k » pour le flip horizontal ça sera la touche « h », pour le flip vertical ça sera la touche « x », pour revenir à l'image originale après le flip ça sera la touche « a » , pour augmenter éclaircir l'image ça sera avec la touche « l » pour l'assombrir ça sera « m » ,pour augmenter la contraste ça sera avec la touche « c » ,pour démineur ça sera « v », pour sauvegarder le resultat d'un effet on rajoute un deuxième argument qui fera office d'image destination au lancement du programme.

2.3.2/ gray.c :

Ce programme marche avec deux paramètre l'image source qui est celle à qui on veut appliquer les modifications et l'image destination qui est l'image source avec l'effet qu'on voulait sur elle ,ce programme utilise la fonction grayscale.

2.3.3/ negatif.c :

Ce programme marche comme gray.c a part qu'il utilise la fonction négatif.

2.3.4/ luminosite.c :

Ce programme marche avec trois arguments l'image source ,l'image destination, l'intensité de la luminosité qui est un int ,il utilise la fonction luminosite et sa fonction auxiliaire fluminosite.

2.3.5/ contraste.c :

Ce programme marche avec trois arguments l'image source ,l'image destination, l'intensité du contraste qui est un double,il utilise la fonction contraste et sa fonction auxiliaire fcontraste.

2.3.6/ flou.c :

Ce programme marche avec trois arguments l'image source ,l'image destination, l'intensité du flou,il utilise la fonction flou et sa fonction auxiliaire moyenne.

2.3.6/ resize.c :

Ce programme marche avec 4 arguments ; l'image source, l'image destination, sa largeur, sa longueur, on charge l'image dans la texture et on affiche la texture dans le rendu, apres on modifie les dimensions de la texture grâce à un rectangle, on lit les pixels du rendu dans une nouvelle surface qu'on sauvegarde.

2.3.6/ flip.c :

Ce programme marche avec trois arguments ; l'image source, l'image destination,un entier pour le type du flip (1 pour un flip horizental , 2 pour un flip vertical),on utilise SDL_Render flip pour le type du flip avec SDL_RenderReadPixels pour afficher le resultat du flip dans une nouvelle surface.

2.3.6/rotate.c :

Ce programme marche avec trois arguments ; l'image source, l'image destination,un double pour le degrés de rotation.

3/bibliographie :

https://openclassrooms.com/courses/apprenez-a-programmer-en-c/installation-de-la-sdl

https://wiki.libsdl.org/

https://www.youtube.com/watch?v=Lwx9rSgwoDg&list=PLrSOXFDHBtfEh6PCE39HERGgbbaIHhy4j&index=23

https://zestedesavoir.com/tutoriels/1014/utiliser-la-sdl-en-langage-c/